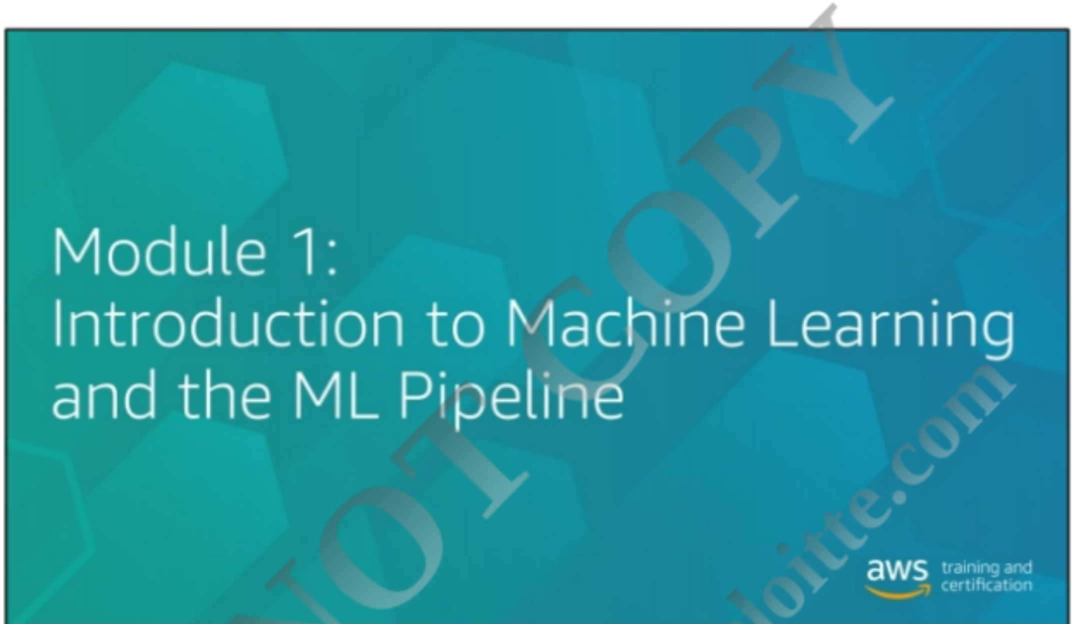
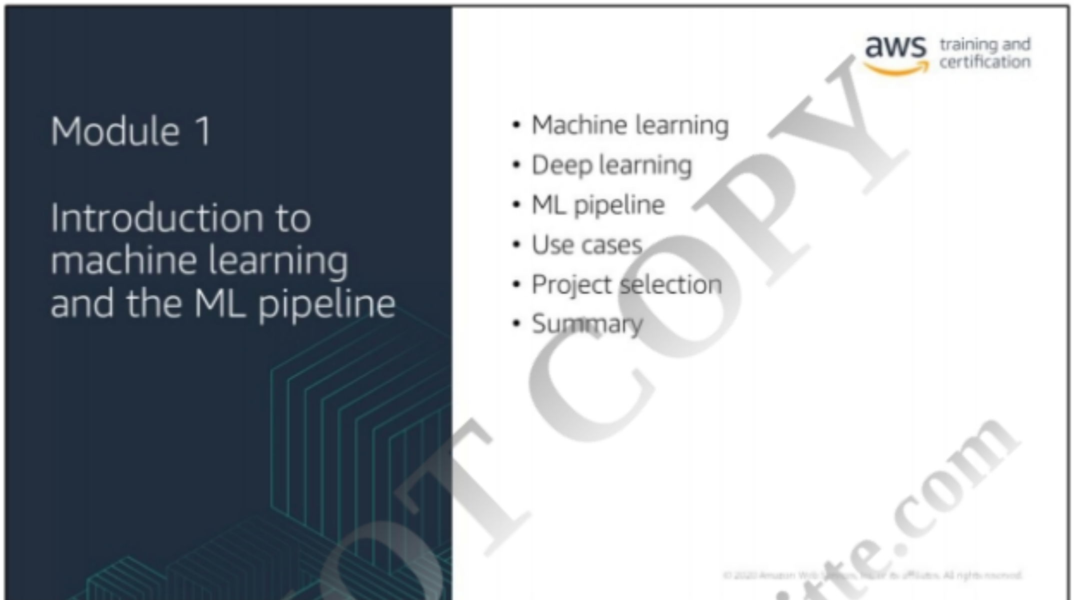
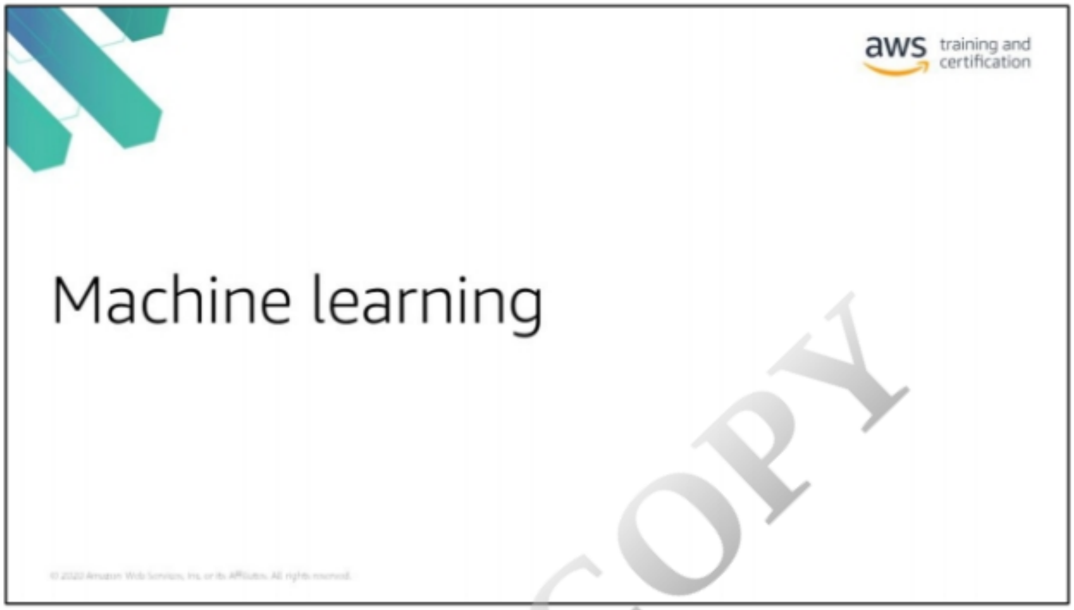
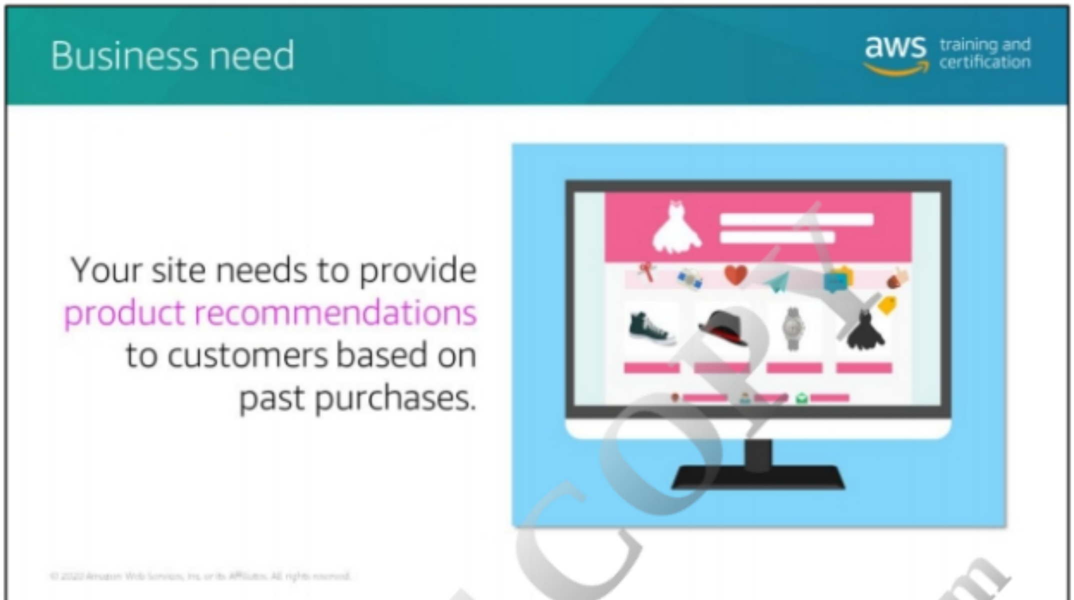

Module 1:
Introduction to Machine Learning
and the ML Pipeline
aws training and certification



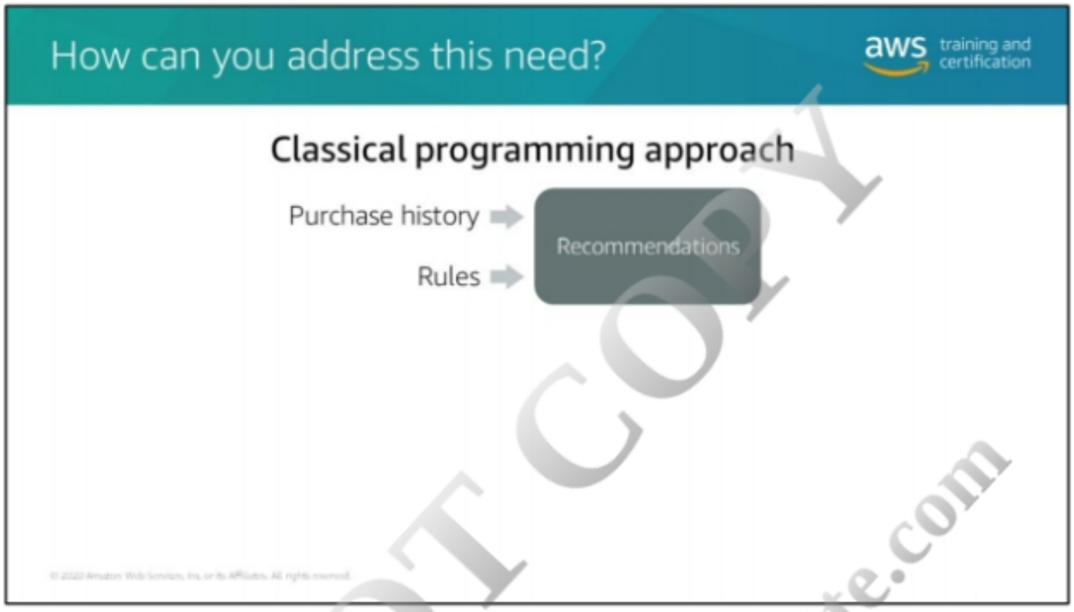
This module is broken into six sections. We'll first introduce machine learning and deep learning, then we'll discuss the phases of the ML pipeline and discus a few use cases. After that you'll learn about three different hands-on projects before selecting one of the projects to focus on the remainder or the course.


aws training and certification
Machine learning



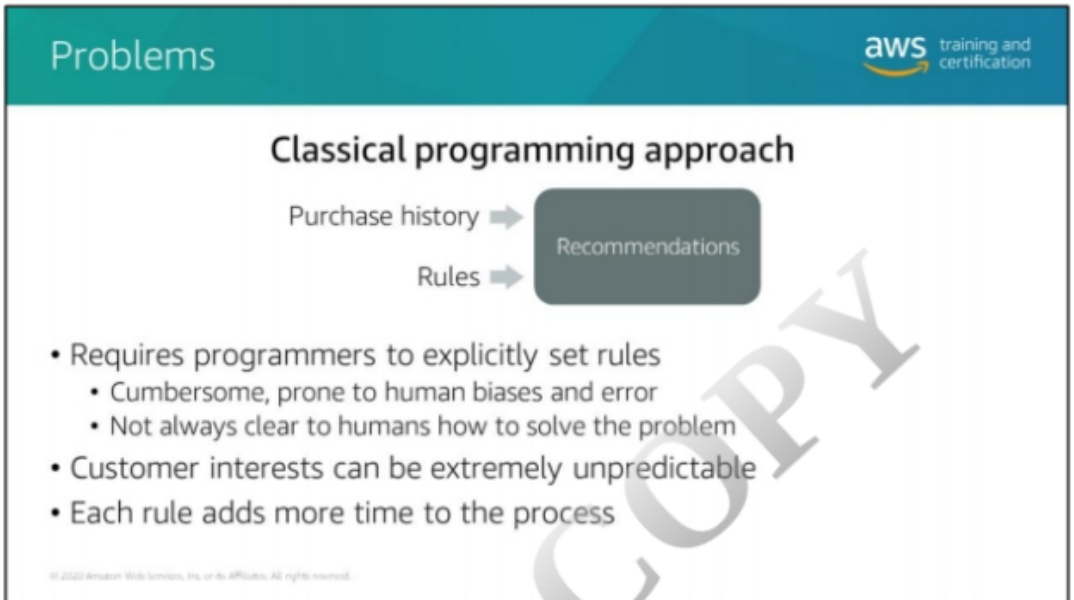
To explain machine learning, let's take a common use case: product recommendations on a shopping site. Let's say you've been tasked with creating the back-end application that will provide product recommendations to customers based on their past purchases.

Image:
https://pixabay.com/vectors/template-layout-website-blog-theme-1599667/#



You could go the classical programming route.

Let's compare machine learning to classical programming. When it comes to developing a prediction (output) from data (input), we know that there needs to be some rules applied to the data. In classical programming, these rules are created by humans, based on factors like business requirements and domain knowledge.

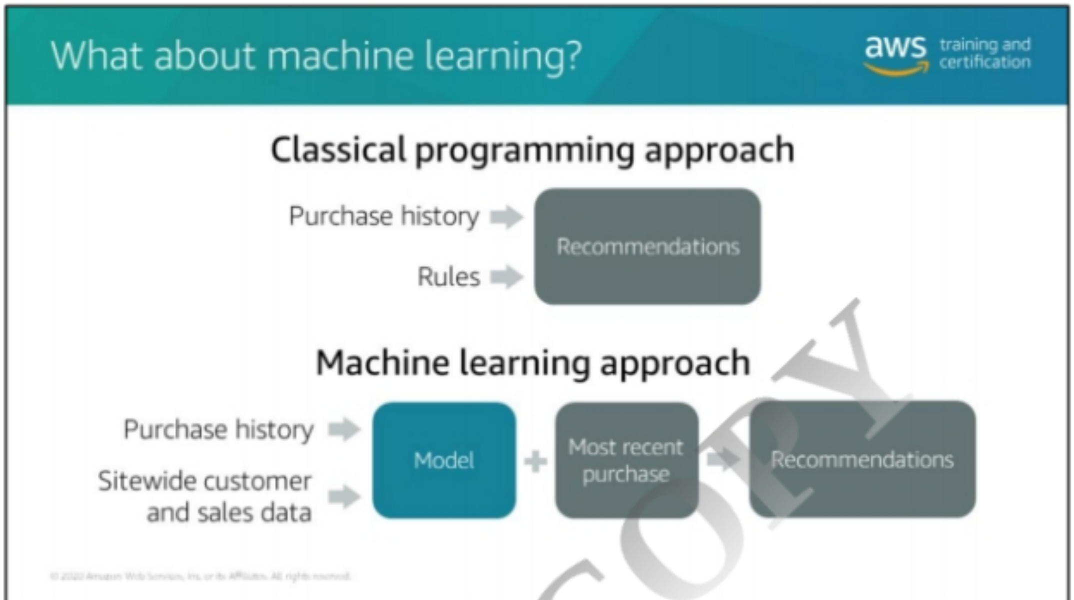


Classical programming is traditionally how these needs were handled in the past.

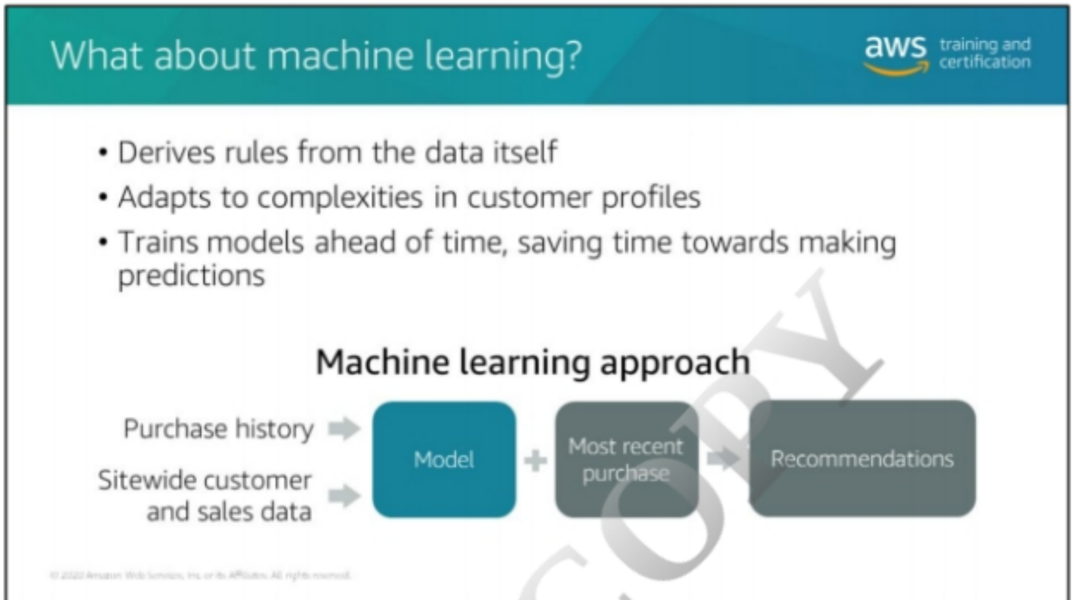
Programmers would set up rules that said, "If this customer purchased product X in the past, show them product Y." because there was some established relationship between those two products. While this can occasionally prompt customers to make that second purchase, it required programmers to explicitly define and set these rules. They couldn't take very much additional context about the customer or the products into account.

In addition, customers are unique and just because one customer was interested in products X and Y, that doesn't mean most or even many customers will be interested in both products as well.

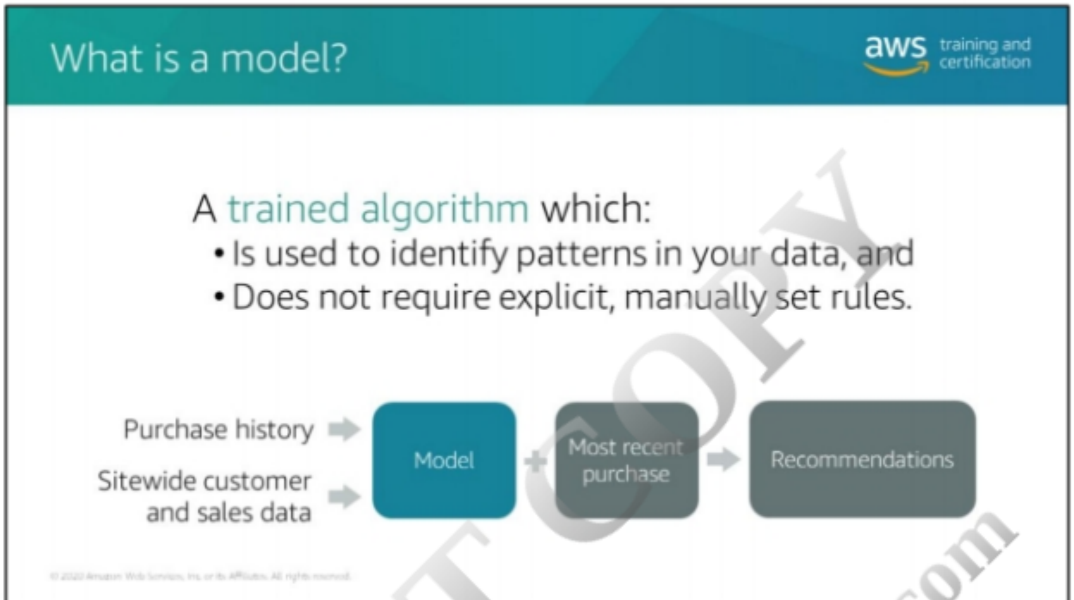
Finally, even if you were to spend the time developing more complex prediction rules, whenever a recommendation needed to be made, the application would have to run through all of the appropriate rules all over again. As more rules are added, the process takes longer to return a result, meaning customers are waiting for those recommendations to load and likely getting frustrated and moving on to another page.



Machine learning, by contrast, would let us use a variety of data collected in the past to automatically derive the patterns hidden in that data. The patterns are then used to create the model, which is applied to new data to provide a more well-informed and adaptive prediction. In this example, we'd be able to use the customer's purchase history in combination with the data of customers and sales sitewide. We can use machine learning to identify the patterns between past customers and sales, then apply those patterns to new customers in order to provide better recommendations to them.



Machine learning, by contrast, would let us use a variety of data collected in the past to automatically derive the patterns hidden in that data. The patterns are then used to create the model, which is applied to new data to provide a more well-informed and adaptive prediction. In this example, we'd be able to use the customer's purchase history in combination with the data of customers and sales sitewide. We can use machine learning to identify the patterns between past customers and sales, then apply those patterns to new customers in order to provide better recommendations to them.



So what is a model, exactly? A model in machine learning is the trained algorithm you use to identify patterns in your data. The key there is that it's trained through the machine learning process. It isn't created manually by programmers setting up rules like in classical programming. Let's look at a very simple example algorithm.

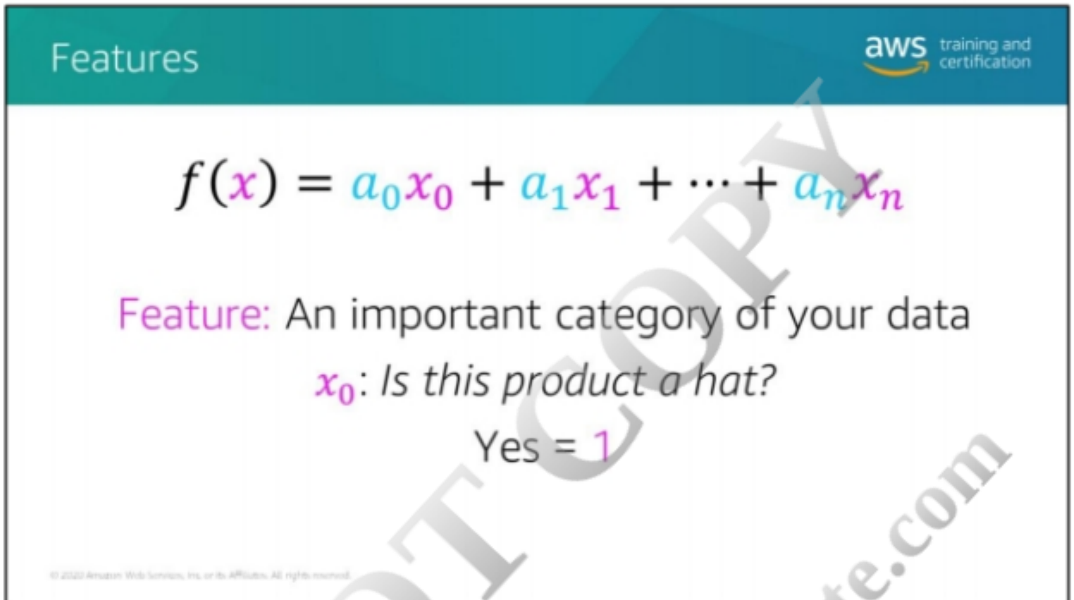
Simple algorithm

aws training and certification

$$f(x) = a_0x_0 + a_1x_1 + \cdots + a_nx_n$$

Features and weights

Here is an example algorithm. This has been simplified from what you might use in a real environment, but it shows the two key components of an algorithm: features and weights.



Features are the parts of your datasets that are identified as important in determining accurate outcomes. For example, with our product recommendation algorithm the first feature might be whether or not the item is a hat. These features have to be expressed mathematically, so in this case our model will convert a Yes into a 1 and a No into a 0.

Weights

aws training and certification

$$f(x) = a_0 1 + a_1 x_1 + \cdots + a_n x_n$$

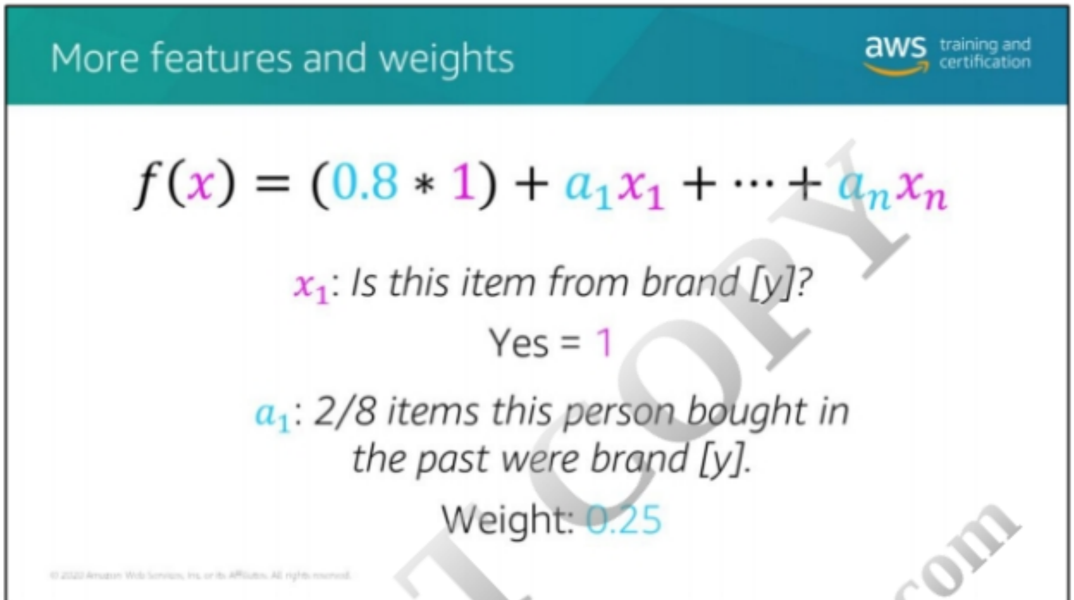
Weight: How much does that feature affect the accuracy of the prediction?

a_0: Hats make up 80% of the purchases this customer has made.

Weight = 0.8

But key to ML is this idea of context. That's where weights come in. Weights represent how important an associated feature is to determining the accuracy of the outcome. So something that has a higher likelihood of accuracy has a higher weight and vice versa. In this case, our model has been trained and has determined that because this customer has purchased 8 hats in the past, that translates to a weight of 0.8.

It's important to note that this is a very simplified version of what really goes on. As the course continues your understanding of weights and features will grow in depth, but for now this is the most critical part of understanding how weights and features work in an algorithm.



Let's look at the next set. The second feature has to do with whether or not a product is from a particular brand. The product is from that brand, so that converts again into a 1. The second weight says that since 2 out of the 8 items this person bought in the past were from that brand, that results in a weight of 0.25.

Output

aws training and certification

If $f(x) > 1$, recommend the product.

$$f(x) = 0.8 * 1 + 0.25 * 1$$

$$f(x) = 1.05$$

If this was a very simple model, this is what you'd end up with. Let's say the standard for whether or not to recommend is if the final result is greater than 1. The model performs this calculation, and finds the final value to be 1.05. Therefore, this product is an acceptable recommendation.

Output

aws training and certification

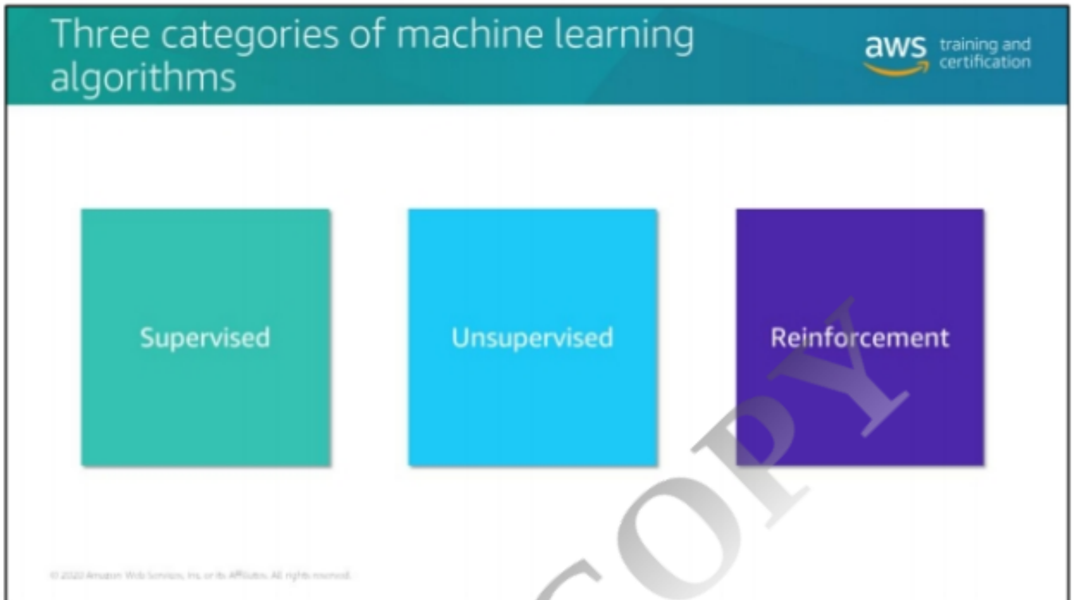
If $f(x) > 1$, recommend the product.

$$f(x) = 0.8 * 1 + 0.25 * 1$$

$$f(x) = 1.05$$

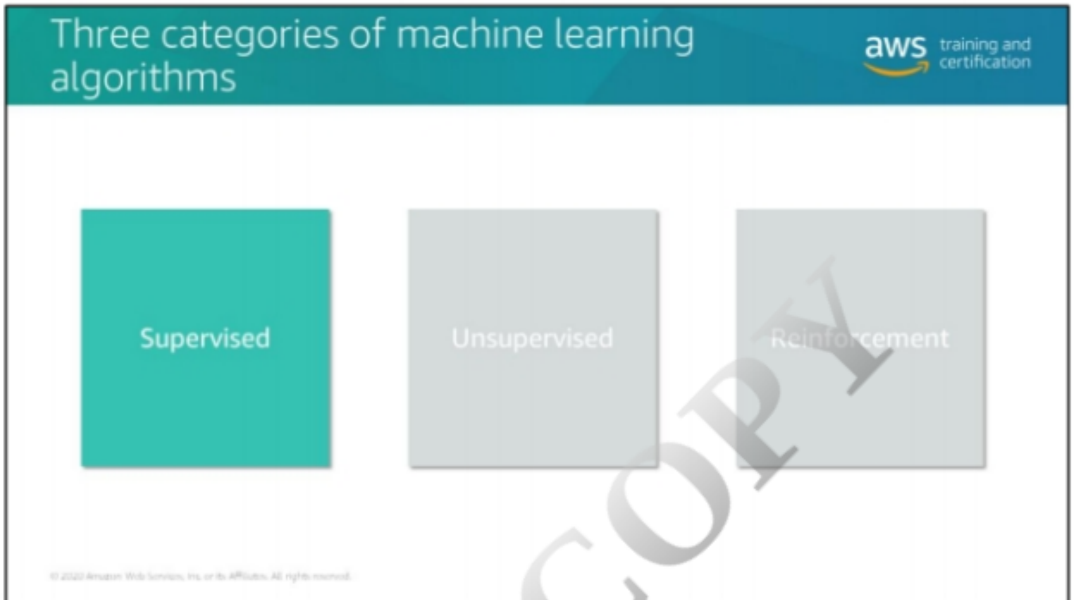
What kinds of algorithms are used?

If this was a very simple model, this is what you'd end up with. Let's say the standard for whether or not to recommend is if the final result is greater than 1. The model performs this calculation, and finds the final value to be 1.05. Therefore, this product is an acceptable recommendation.



Let's get back to the different types of machine learning algorithms. There's *supervised learning* where a model uses known inputs and outputs to generalize future outputs. There's *unsupervised learning* where the model doesn't know inputs or outputs so it finds patterns in the data without help. There's *reinforcement learning* where the model interacts with its environment and learns to take actions that will maximize rewards. And then there's *deep learning* which is a subset of machine learning. These algorithms learn by using artificial neural networks, which we'll talk about in a bit.

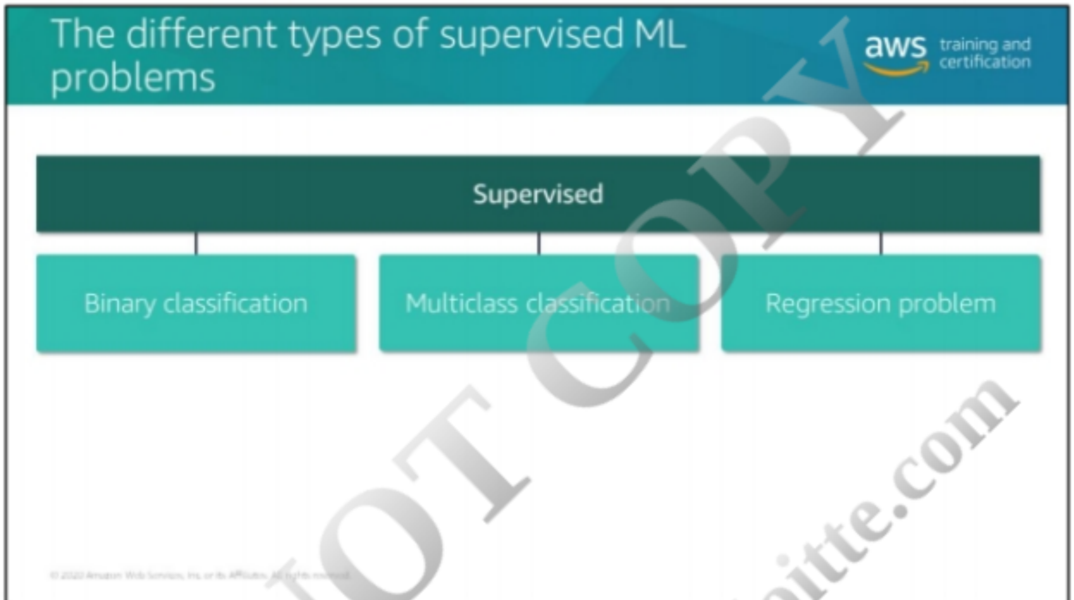
Again, it's important to know the types of ML because the type will guide you toward selecting an algorithm(s) that makes sense for solving your business problem.



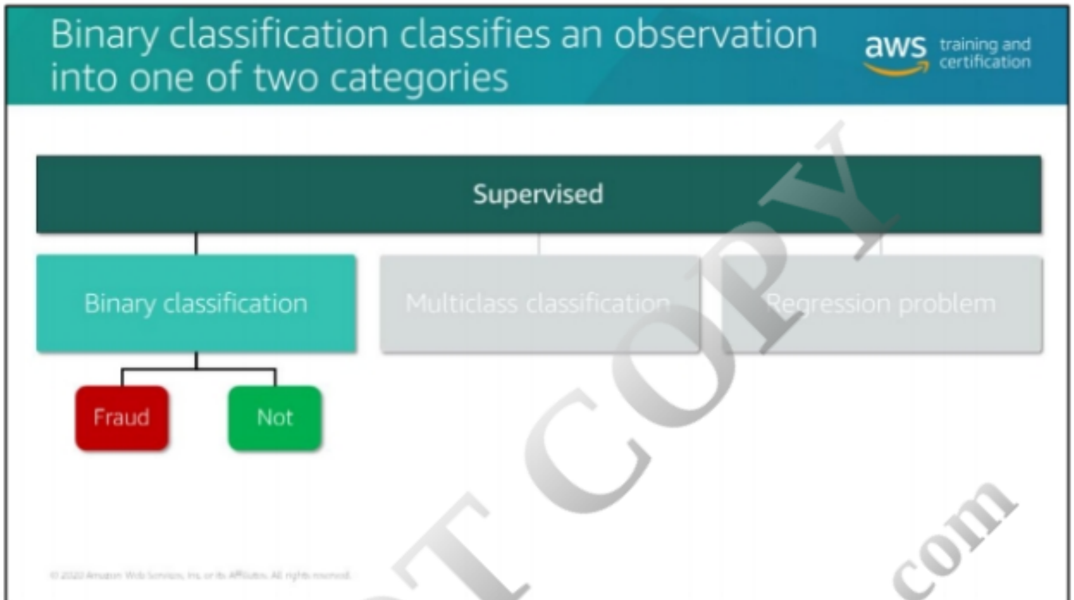
First let's talk about supervised machine learning.

Supervised learning is a popular type of ML because it's widely applicable. It's called supervised learning because there needs to be a supervisor – a teacher who can show the right answers, so to speak.

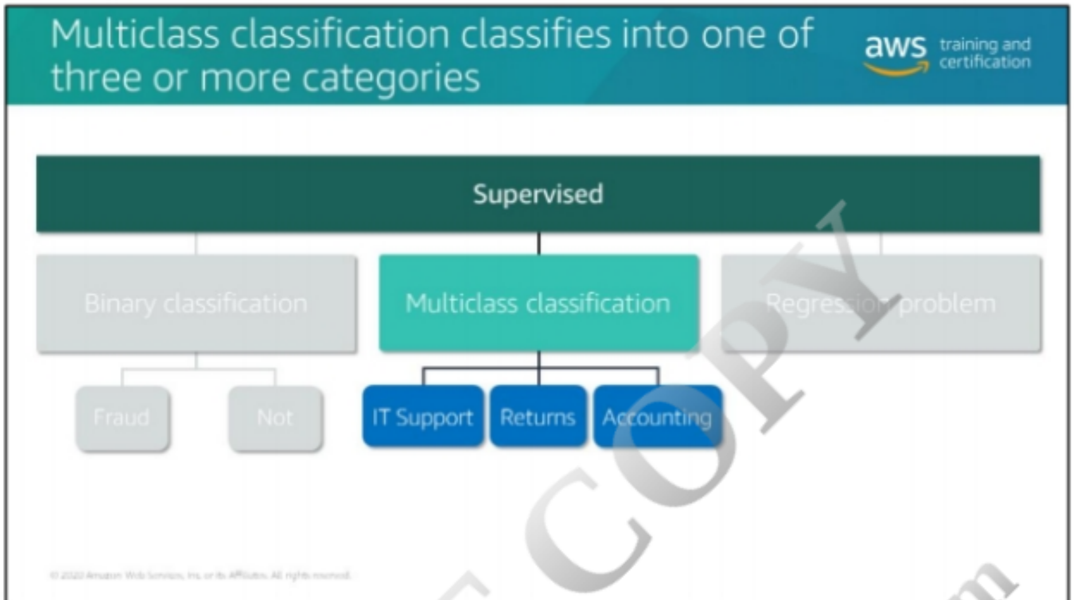
Like any student, a supervised algorithm needs to learn by example. Essentially, it needs a teacher who uses training data to help it determine the patterns and relationships between the inputs and outputs. "Here in this picture is a car. Here is a car in another picture." The model is trained on this labeled data so that it can accurately identify where a car is in a new picture it hasn't seen before.



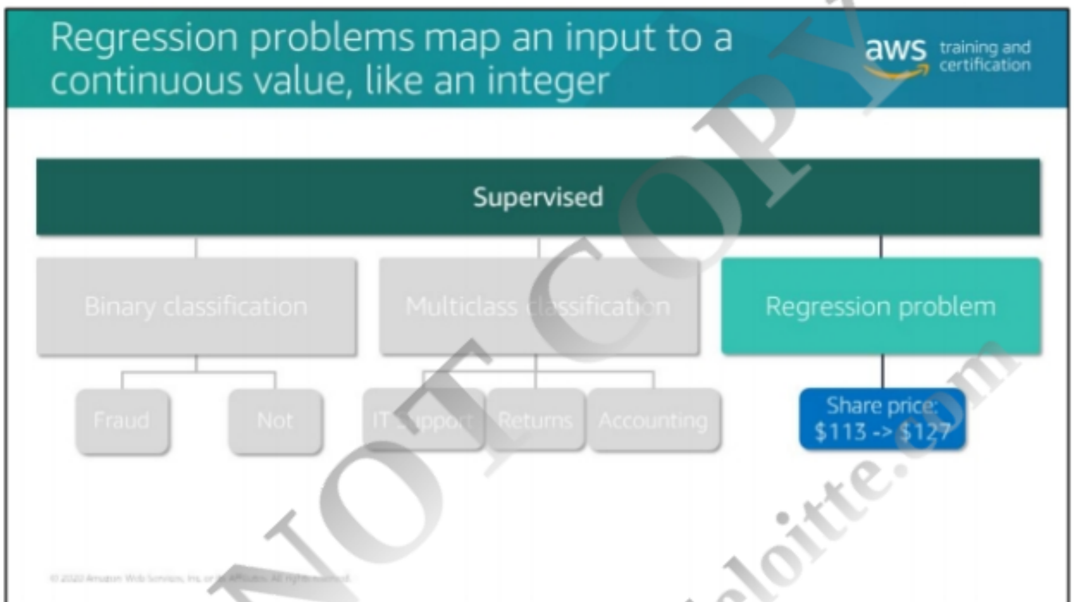
But within supervised learning, you have different types of problems. These can be broadly categorized into two categories, classification and regression. Think of ML problems falling into one of two categories: classification and regression.



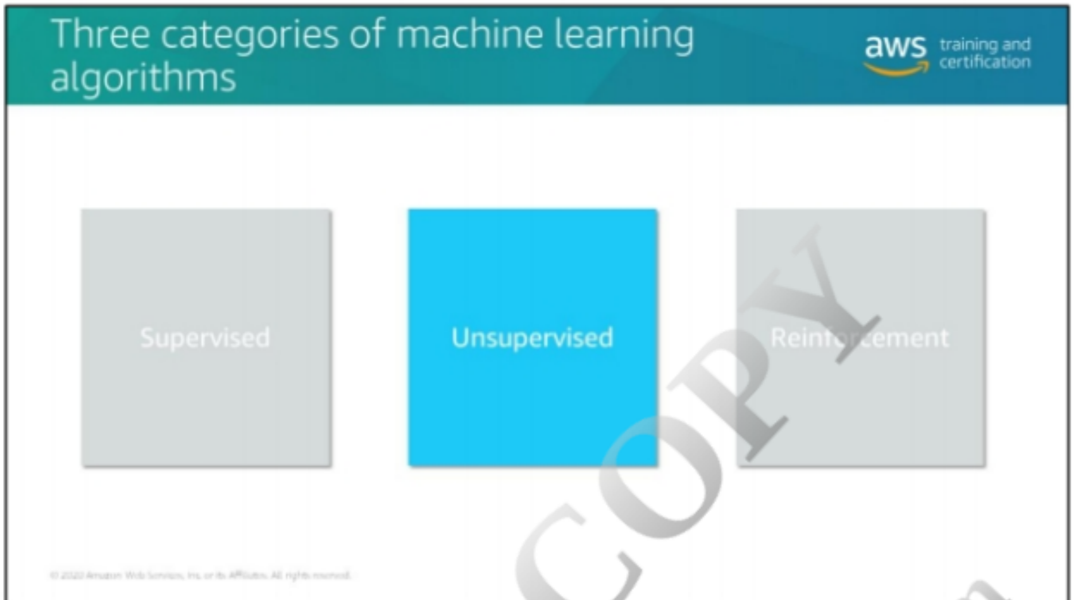
Within *classification* problems, there are actually two types. The first is considered a binary classification problem. Think back to the example provided above about identifying fraudulent transactions. The target variable in this example is limited to two options: fraudulent or not fraudulent. This is an example of a binary classification problem: you are classifying an observation into one of two categories.



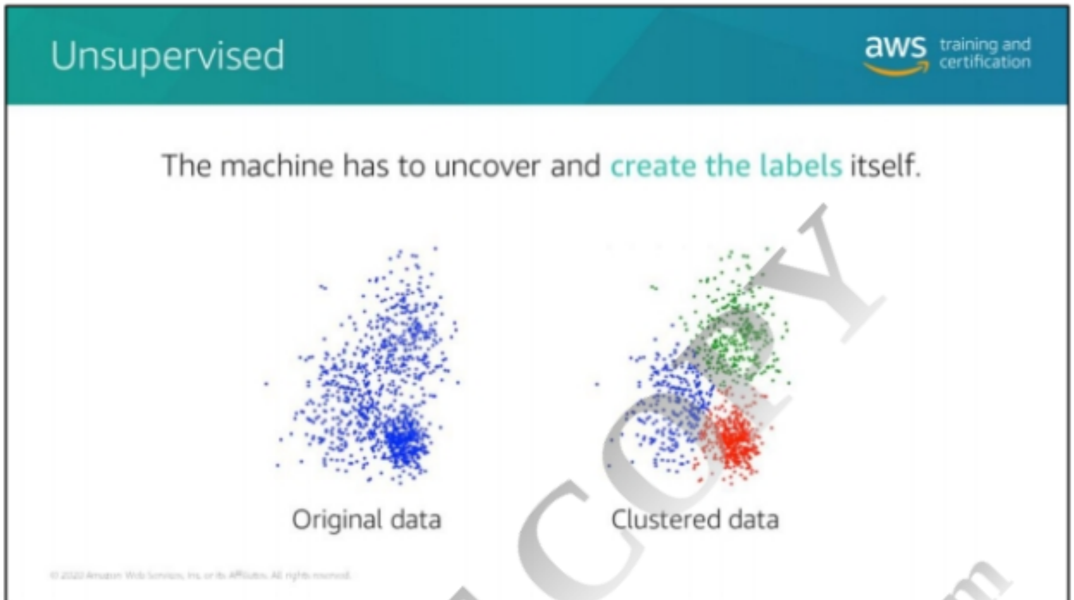
There are also multiclass classification problems. These ML problems classify an observation into one of three or more categories. Pretend you have an ML model that predicts why a customer is calling your store so that you can reduce the number of transfers needed before getting the customer to the right customer support department. The different customer support departments, in this case, represent the variety of potential target variables—which, needless to say, could be many different departments, far greater than just two.



There are also *regression* problems. In a regression problem, you're no longer mapping an input to a defined number of categories, but instead to a continuous value, like an integer. One example of an ML regression problem is predicting the price of a company's stock, for example, here a regression-based algorithm is predicting that tomorrow, the stock price for a company will go up from $113 per share to $127 per share.



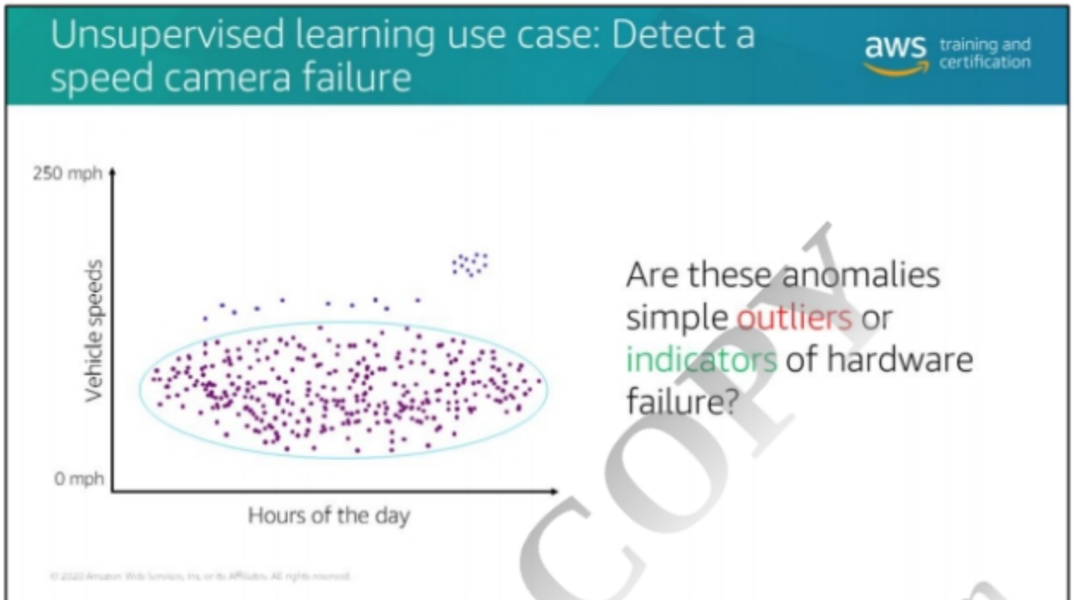
Now let's talk about unsupervised machine learning.



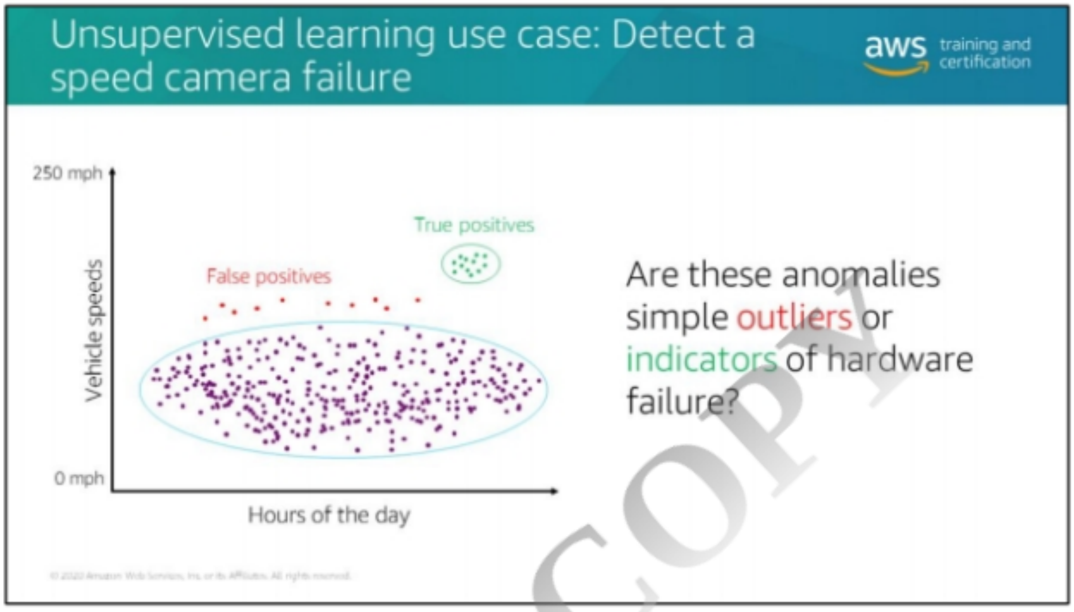
Sometimes, all we've got is the data – there's no supervisor in the room. In unsupervised learning, labels are not provided like they are with supervised learning – we don't know all the variables and patterns. In these instances, the machine has to uncover and create the labels itself. These models use the data they're presented with to detect emerging properties of the entire dataset, and then construct patterns.

A common subcategory of unsupervised learning is called 'clustering'. This kind of algorithm groups data into different clusters based on similar features, in order to better understand the attributes of a specific cluster. For example, by analyzing customer purchasing habits, unsupervised algorithms are capable of identifying groups of customers that identify a particular company as being large or small. It may be sufficient for smaller companies to purchase basic cloud hosting resources, while larger companies may be more likely to purchase entire cloud solutions including advanced security, dedicated private connections, virtual private clouds, and more. Clustering in this situation might help you realize that you need to come up with a different marketing strategy for different sized companies.

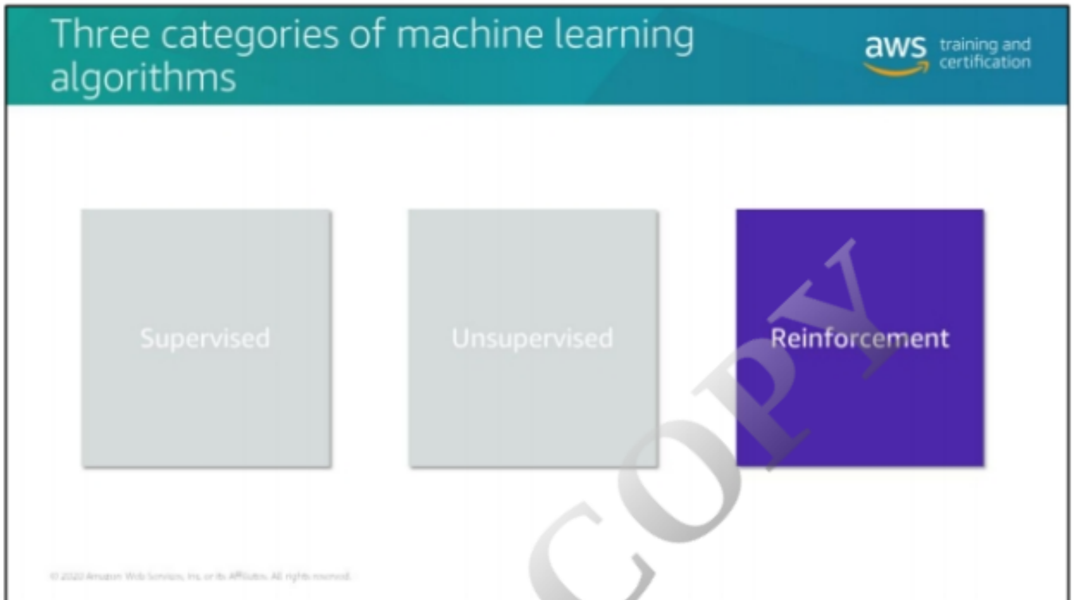
The advantage of unsupervised algorithms is they enable you to see patterns in the data that you were otherwise unaware of – like the existence of two major customer types.



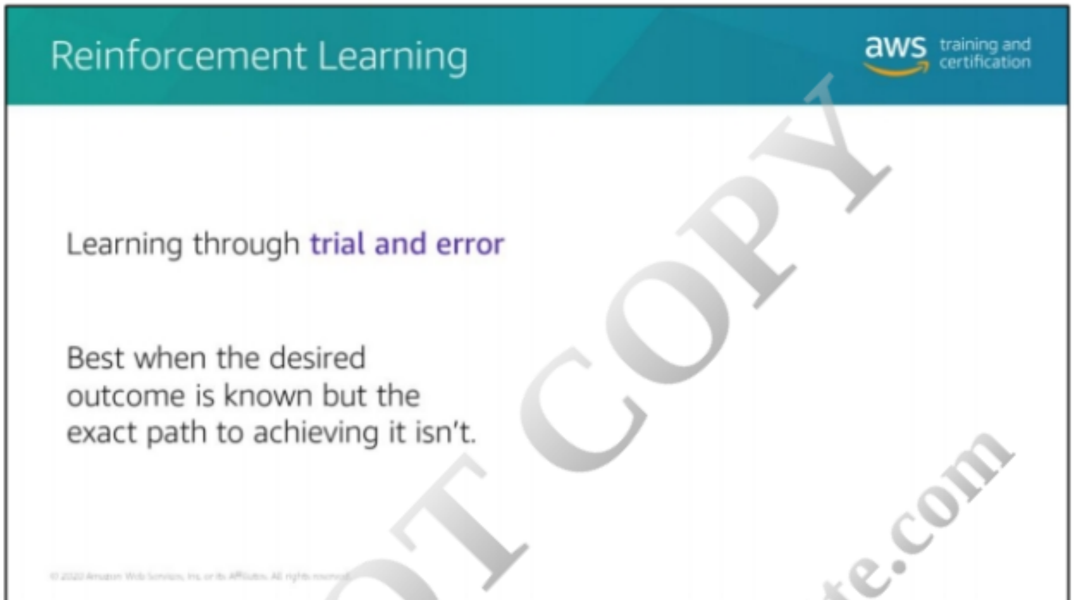
One use case for unsupervised learning is anomaly detection, such as the speed camera examples above. This model is designed to detect potential hardware failures of speed cameras by looking for anomalies in the data. In a classical programming solution, every datapoint outside of a clearly defined boundary would have to be evaluated as a potential failure. But with unsupervised learning, models can identify what are simple outliers (such as people driving at very high speeds at random points during the day) and what is more likely to be a result of hardware failure (a high volume of extremely high speeds recorded over a span of time).



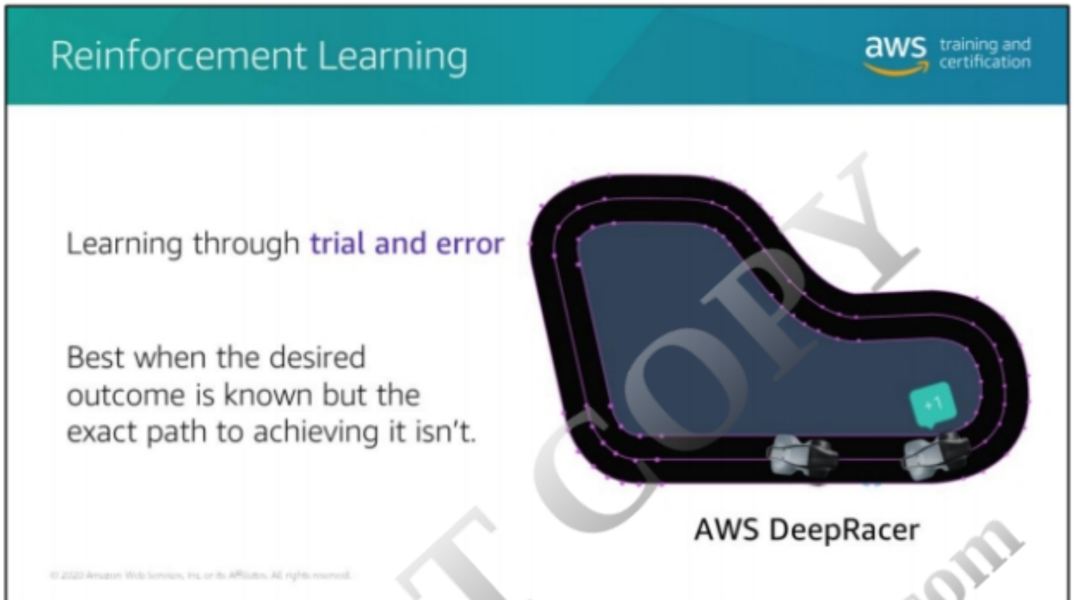
One use case for unsupervised learning is anomaly detection, such as the speed camera examples above. This model is designed to detect potential hardware failures of speed cameras by looking for anomalies in the data. In a classical programming solution, every datapoint outside of a clearly defined boundary would have to be evaluated as a potential failure. But with unsupervised learning, models can identify what are simple outliers (such as people driving at very high speeds at random points during the day) and what is more likely to be a result of hardware failure (a high volume of extremely high speeds recorded over a span of time).



Now let's talk about reinforcement learning.

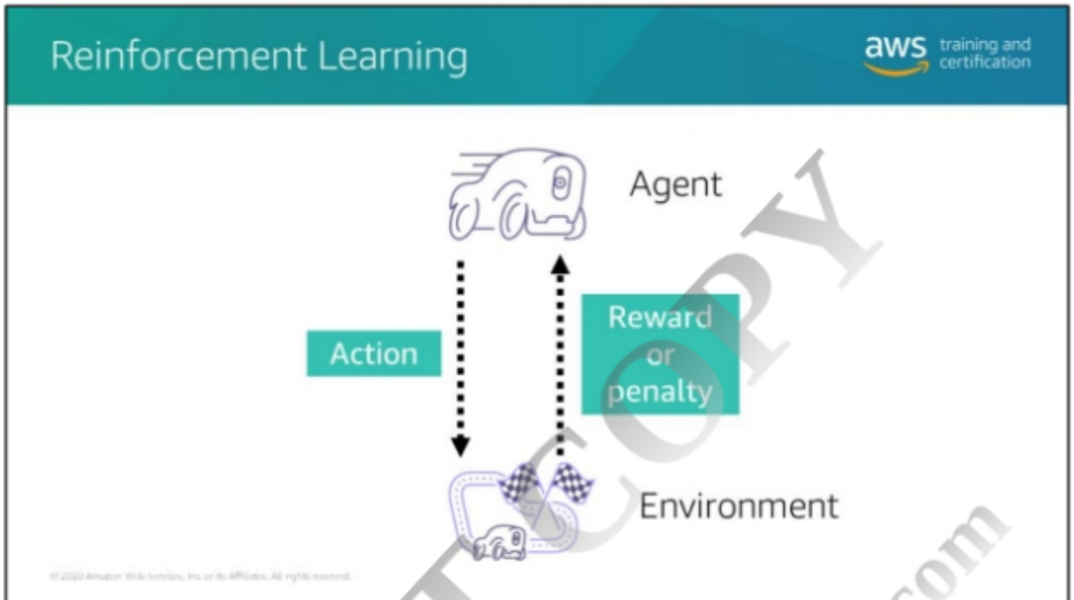


Another kind of algorithm that's been gaining in popularity recently is reinforcement learning. Unlike the first two algorithms this one continuously improves its model by mining feedback from previous iterations. In reinforcement learning, an agent continuously learns, through trial and error, as it interacts in an environment. Reinforcement learning is broadly useful when the reward of a desired outcome is known but the path to achieving it isn't – and that path requires a lot of trial and error to discover.



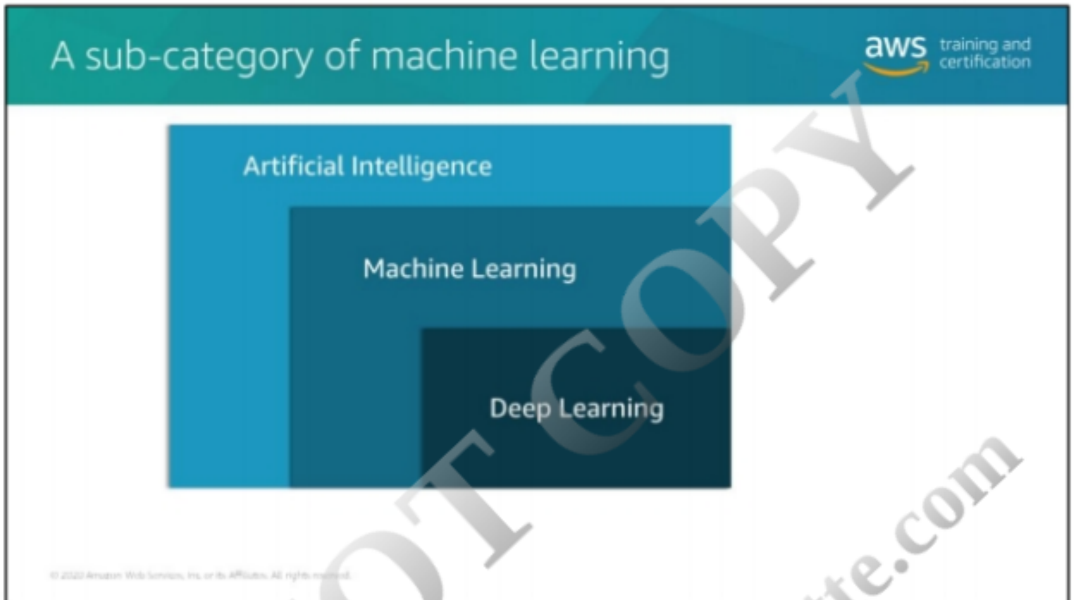
Take the example of AWS DeepRacer as shown on the slide. In the AWS DeepRacer simulator the agent is the virtual car, the environment is a virtual racetrack, the actions are throttle and steering inputs to car, and the goal is completing the racetrack as quickly as possible and without deviating from the track.

The car needs to learn the desired driving behavior to reach our goal of completing the track. To learn this, we will use rewards to incentivize our model to learn the desired driving behavior.

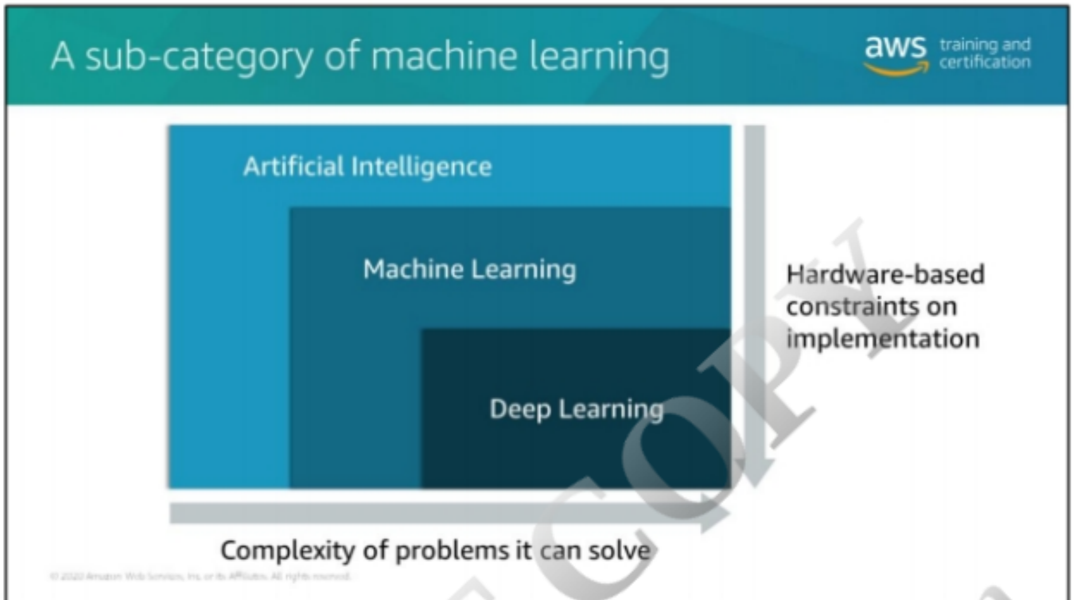


In reinforcement learning, the thing driving the learning is called the agent. In this case, it's the DeepRacer car. The environment is the place where the agent learns, which in this example would be the marked race track. When the agent does something in the environment that provokes a response, such as crossing a boundary it shouldn't cross, that's called an action. That response is called a reward or penalty depending on whether the agent did something to be reinforced or discouraged in the model. As the agent moves within the environment, its actions should start receiving more and more rewards and fewer and fewer penalties, until it meets the desired business outcome.

Deep learning



Now that we've talked about the 3 categories of machine learning, let's talk a little about deep learning. Deep learning (or DL) is a subset of machine learning. These algorithms learn using artificial neural networks – let's dig into that a little.

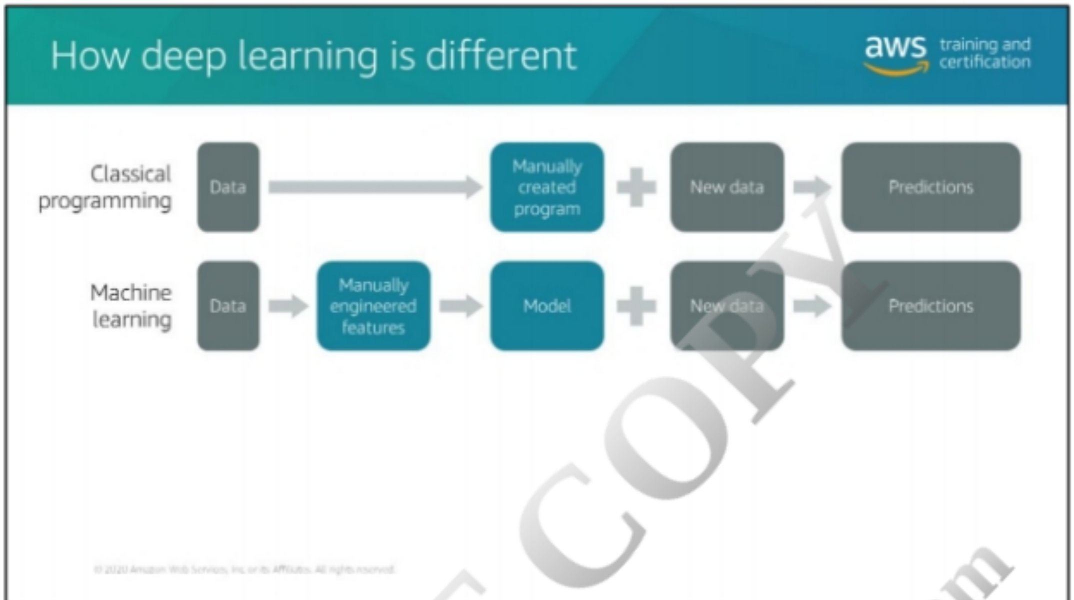


Deep learning represents a huge leap forward in the capabilities for artificial intelligence and machine learning. While the theory for deep learning goes back decades, the hardware required to run deep learning problems wasn't generally accessible until very recently. But now that it's available, we're able to use deep learning to address problems more complex than we could before.

To demonstrate the leaps between these three categories, we can use the classic example of machines that can play chess. With classic artificial intelligence, the machines couldn't learn anything that wasn't given to them directly in the code. So a machine could "play" chess based on rules that were provided, but any improvements to its strategy would have to come from a programmer explicitly calling them out and programming them in.

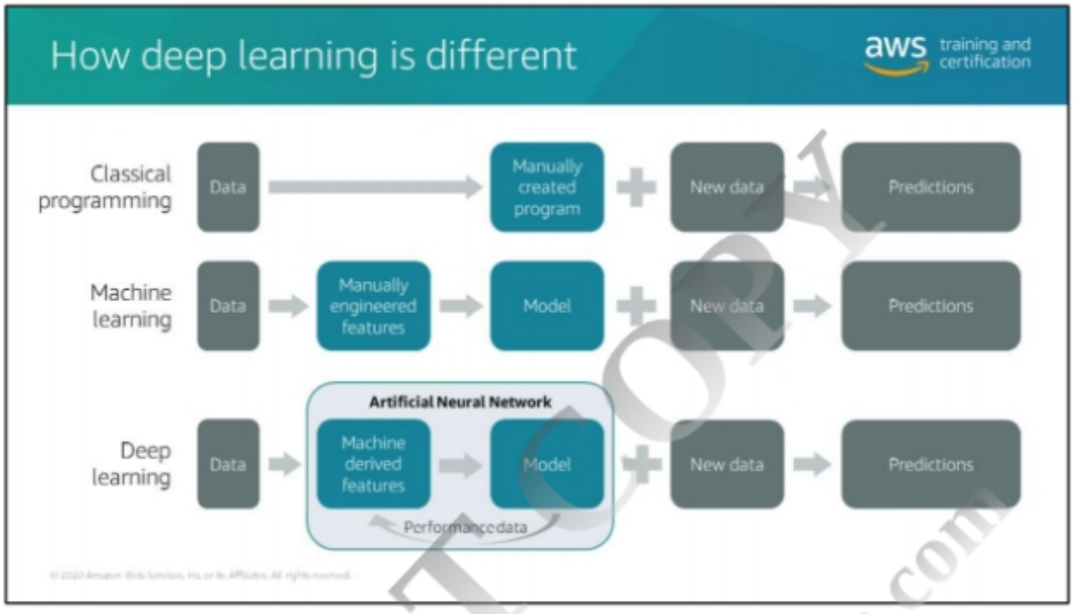
With machine learning, as we've learned in this module, chess-playing machines can actually learn and improve their chess skills by doing things like analyzing past chess games that were played by humans. They could be provided with simple chess strategies, but then use something like traditional reinforcement learning, where they can identify patterns in moves that were more likely to result in winning versus moves that were more likely to result in losing the game, to hone their performance.

But now with deep learning, we are able to create machines that can learn chess in a fashion more like how humans learn it: by playing. A deep learning machine could be given the most basic rules on how chess is played, and use its complex, layer-based, iterative approach to learn its own chess strategies and recognize patterns several orders of magnitude more complex than a traditional machine learning model could.

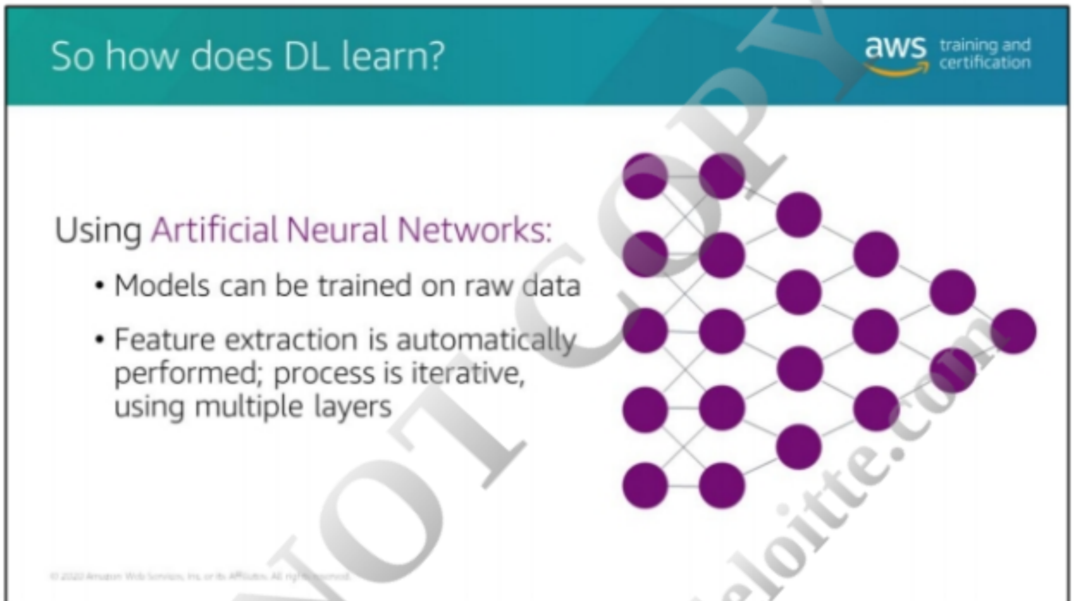


We've talked about features, their importance, and how they're used in classical programming. Let's look at how the use of features evolved from classical programming, to ML, to DL.

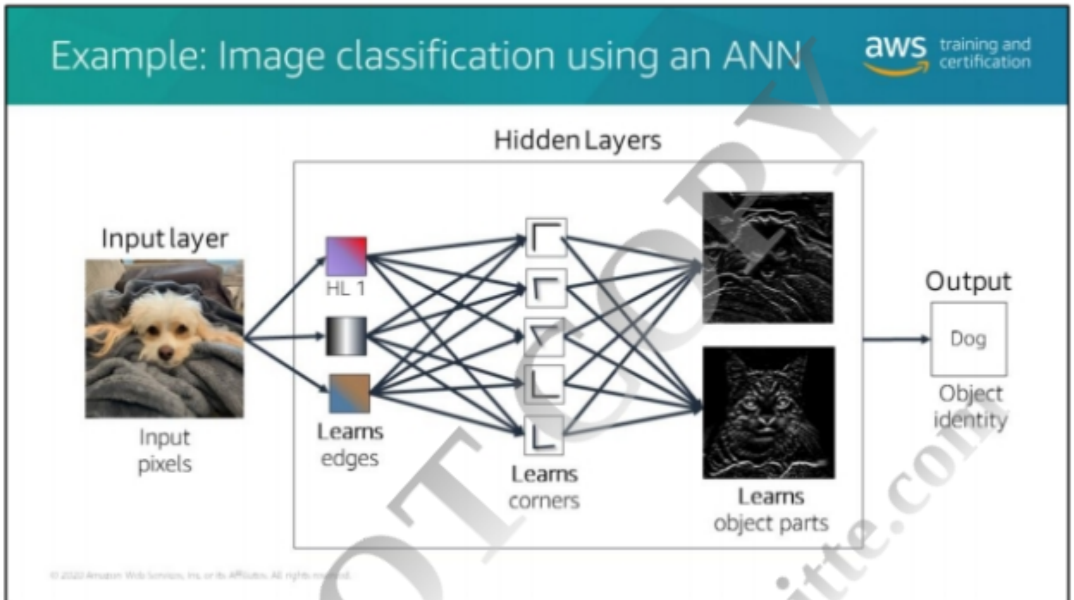
As we've established before, in machine learning, the features are manually engineered, and then used to train and develop the model.



But in deep learning, the features are derived by the ANN itself during the training and tuning process. The performance of the model based on that training is then fed back in through the ANN iteratively until generating the final model.

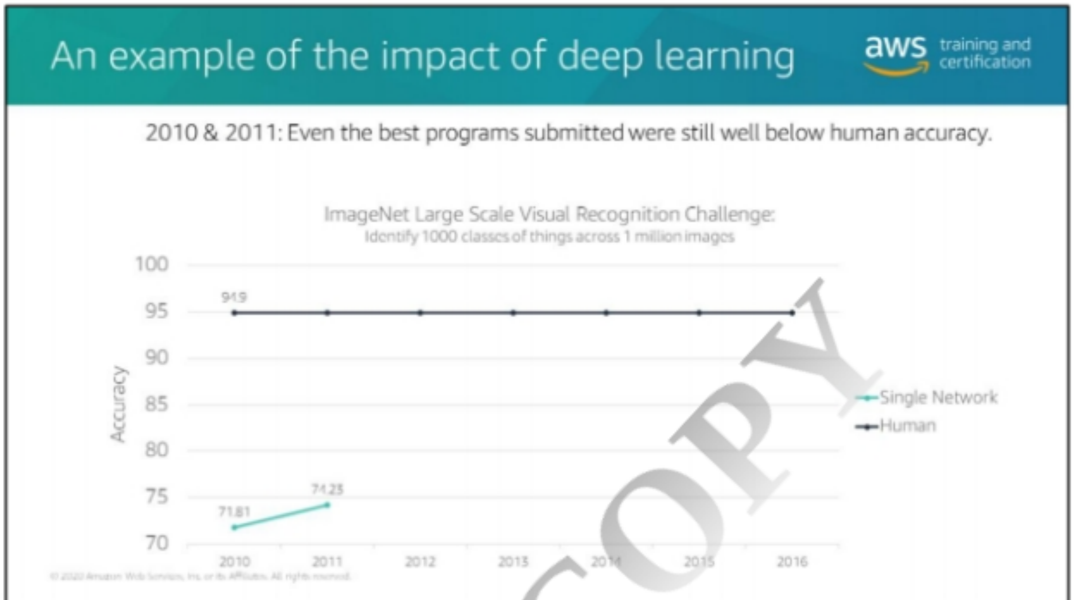


The deep learning models in use today rely on things called artificial neurons. These are math functions that receive inputs and sum them up to create an output. Work on artificial neurons began back in the 40's, however, it wasn't until technological breakthroughs in the 2010s enabled Artificial Neural Networks (ANN) to be used in real environments. In an ANN, each layer summarizes and feeds information to the next layer, ultimately producing a final output or prediction. During this process, the model derives the features itself.



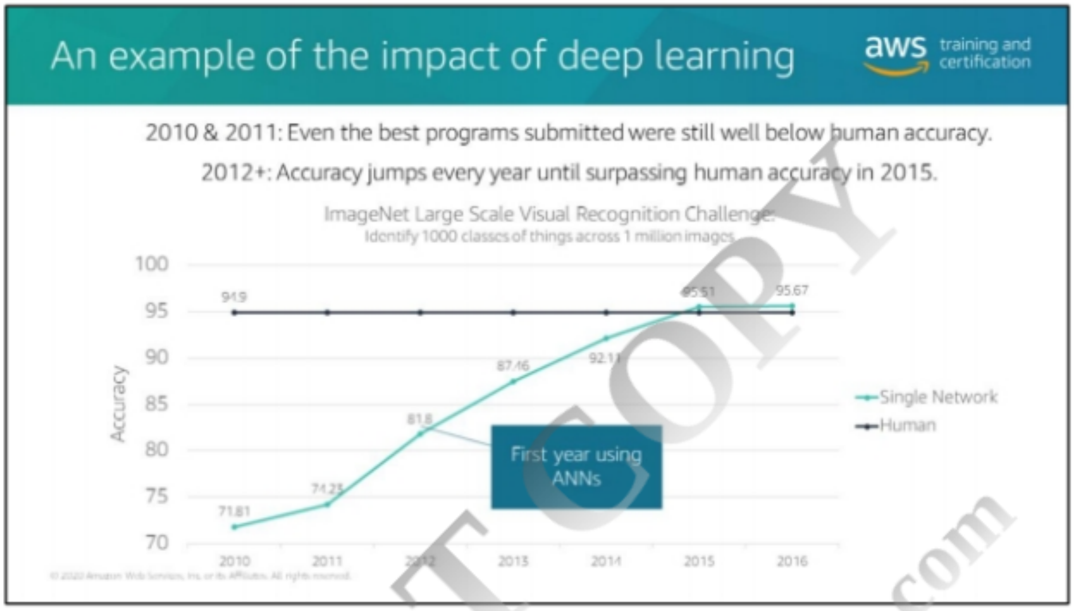
Let's break this down into a more concrete example of a deep learning analysis of an image. In this example, we see that the machine starts by identifying specific features of the input beginning with pixels. From there it identifies edges, corners and contours, and object parts before making a prediction as to what the object is. In this way it is extracting features, like edges, corners, and parts automatically based on the algorithm and the data it's being trained on. It can identify patterns at levels the human eye can't easily identify.

Example: Image classification using ANN



Again, the premises for deep learning started in the early 20th century but the impact of improvements made in the field was really felt in the early 2000's. The big shift began with the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) which was a competition where research teams submitted programs that classified and detected objects and scenes in photographs.

- In **2010**, the first year of the challenge, the winner of the challenge had a model with an accuracy rate of 71.81%.
- In **2011**, accuracy improved marginally to around 74%.



Back to the ImageNet Large Scale Visual Recognition Challenge
- In **2012**, when deep convolutional neural networks, common in deep learning models, started to be used, the accuracy rate jumped up to about 82% and has been steadily climbing ever since.
- Then in **2015**, the winning program actually exceeded human performance at these narrow ILSVRC tasks.

Deep learning recap

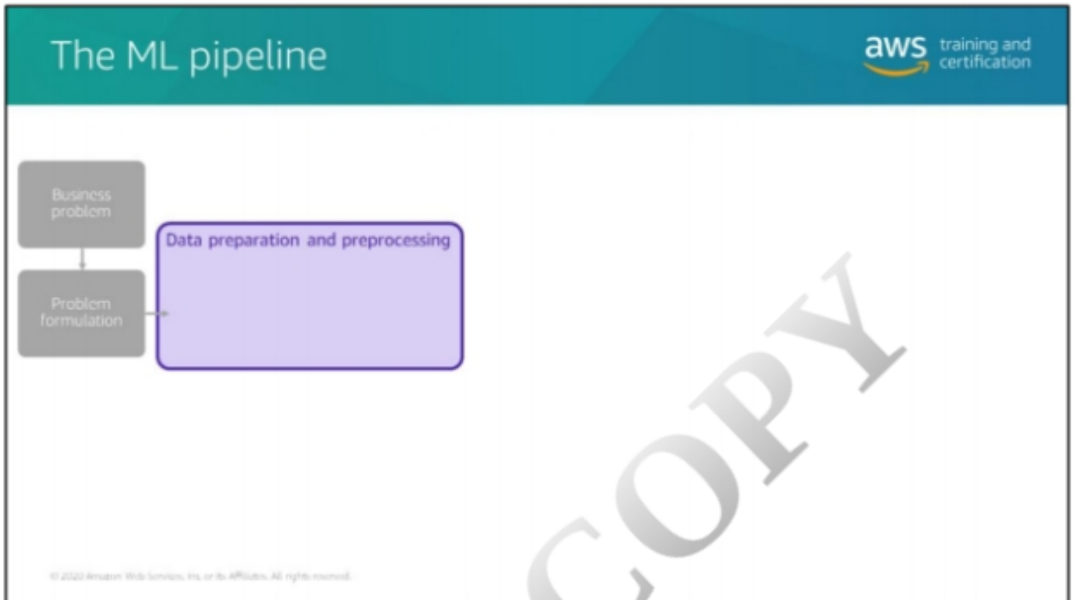
aws training and certification

Deep learning models

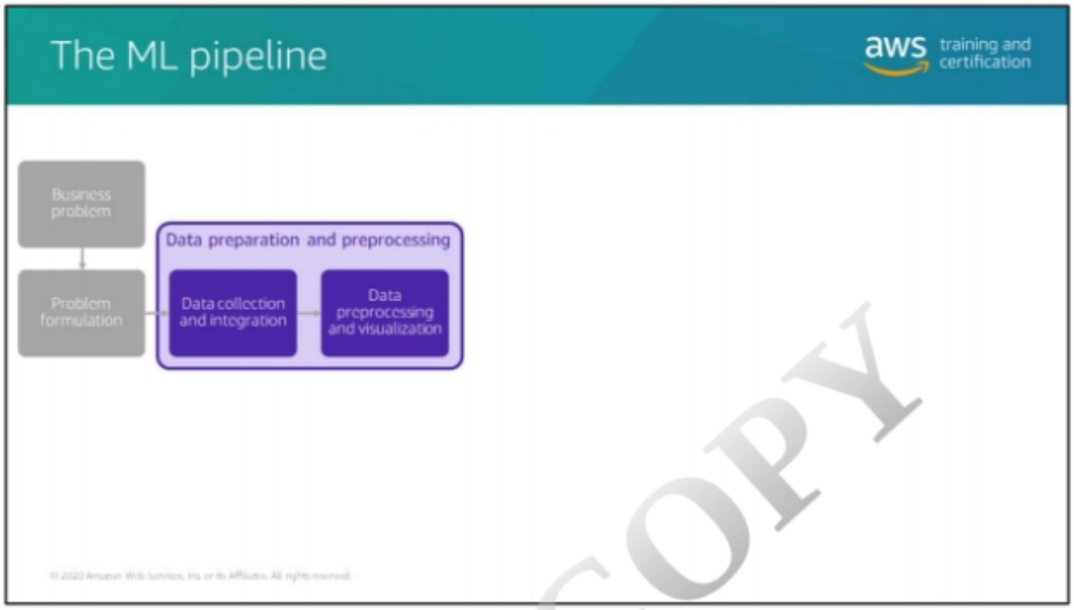
- Learn using Artificial Neural Networks (ANN)
- Can be trained on raw features
- Include:
  - A subset of ML methods
  - Neurons arranged for computational efficiency
  - Training networks with hundreds of layers that learn and improve the features


aws training and certification
ML pipeline

So to begin, you should always start with the business problem you or your team believe could benefit from machine learning. From there, you want to do some problem formulation. This phase entails, in part, articulating your business problem and converting it to an ML problem. Again, we'll get into more detail later on, but for now let's just talk generally about what's going on.



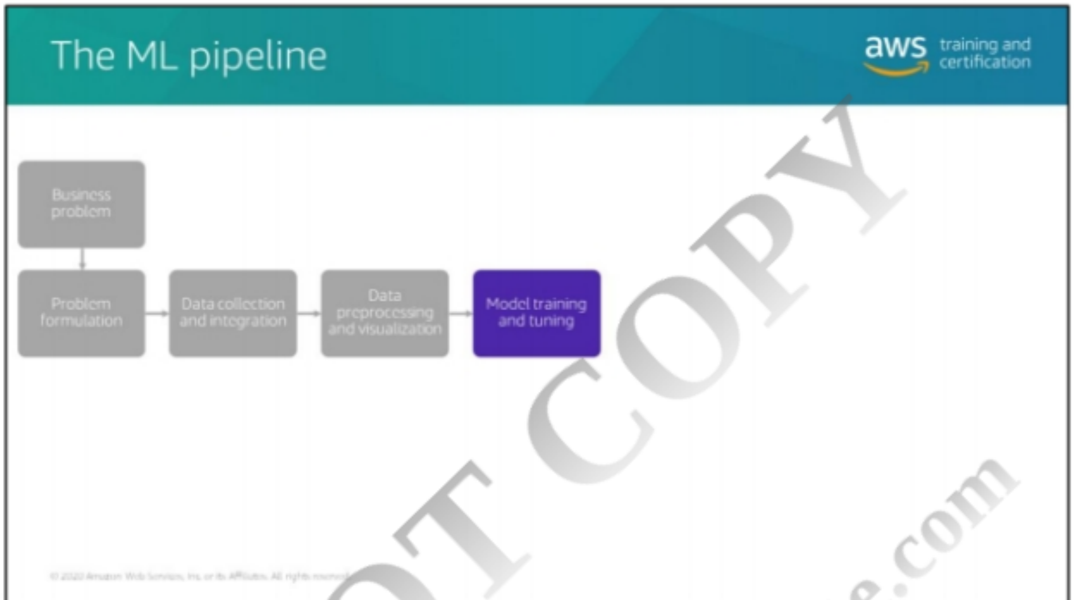
After you've formulated the problem, you move on to the data preparation and preprocessing phase.



Data preparation and preprocessing includes the following:

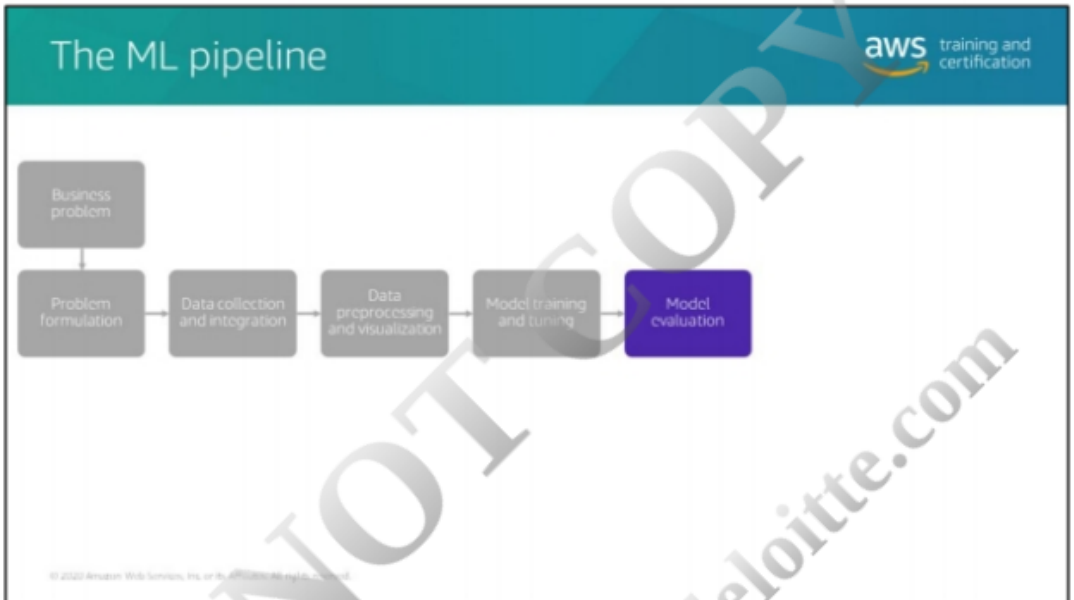
- Data collection and integration in order to ensure your raw data is in one central, accessible place
- Data preprocessing and data visualization which involves transforming raw data into an understandable format and extracting important features from the data

All this is done to ensure your data is formatted correctly for your ML algorithm and that your data is cleaned up in a way that will maximize your model's prediction power.

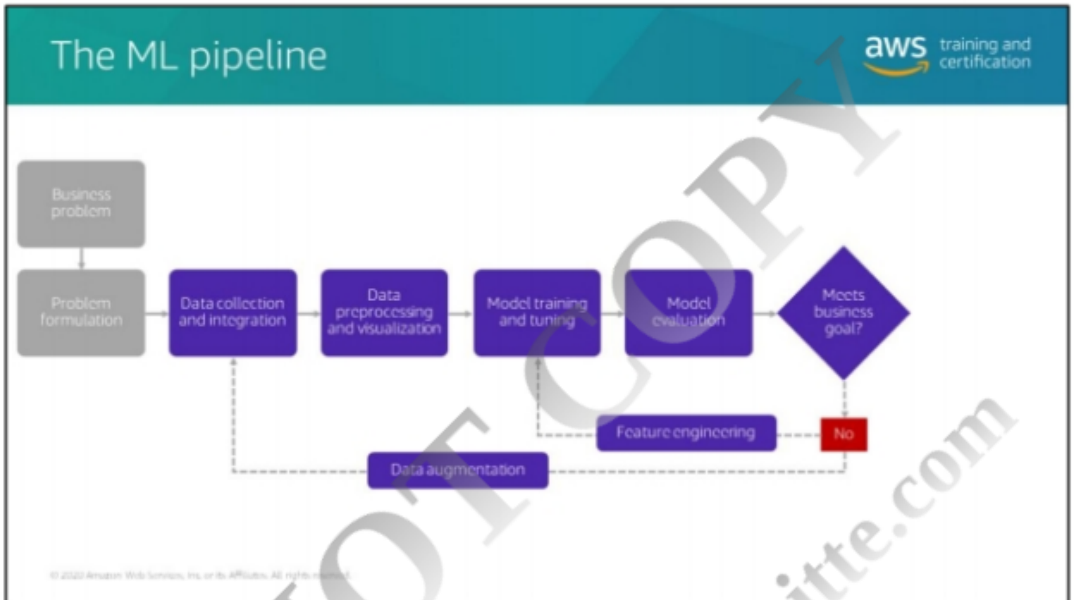


Once those steps are complete, you are ready to train and tune your model. This is an iterative process that can be performed many different times throughout this workflow.

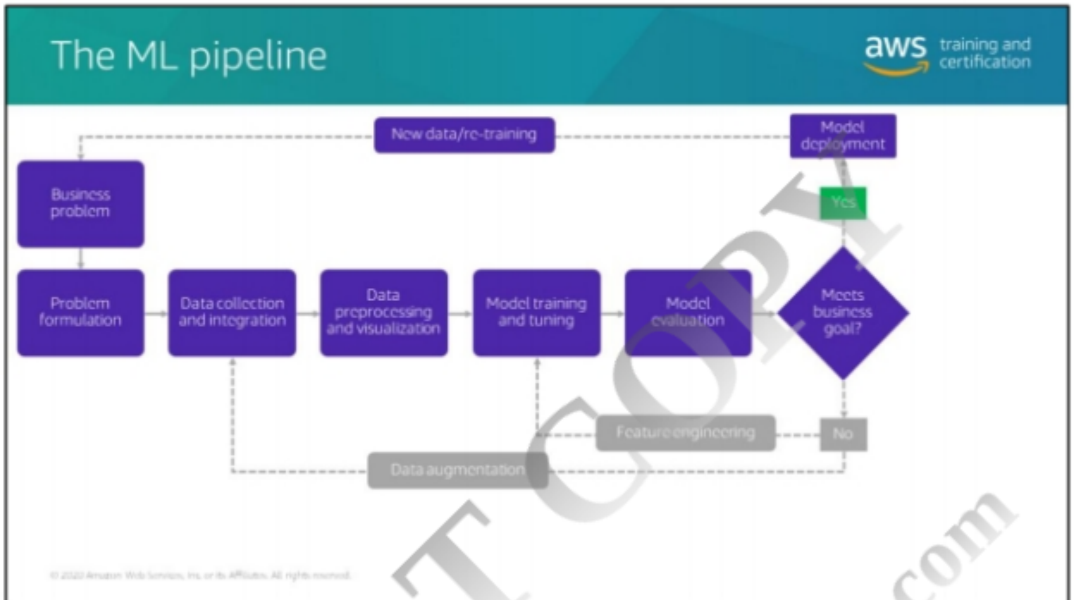
Initially, upon training, your model will not yield the results you might be expecting. Therefore, you will perform additional feature engineering and tune your model's hyperparameters before retraining.



This cycle is repeated until your model's evaluation shows it is performing at the level required by your business case.



If your model *doesn't* meet your business goals, you will need to go back and re-evaluate a few things. Take a second look at your data and features for ways to improve the model, and the way that it's produced. Building a model usually is an iterative process in this way.



Once the retraining happens and you're satisfied with the results, your model is deployed to deliver the best possible predictions, which is often a tedious and manual effort.

The bulk of this course, is designed to walk you through these different phases and to give you hands-on experience with each of them. Shortly, you'll choose a project that you will take through this workflow.


aws training and certification
Use cases